Claims

1. A static error checking system for analyzing a software system in order to detect design errors prior to system execution, the software system comprising a set of software elements which expose control interactions between the software elements, by representing the control interactions in a control graph, the control graph comprising:

a set of conjunctive nodes, each of which represents a conjunctive boolean guard on state changes within the software system;

a set of disjunctive nodes, each of which represents a boolean guard on a functional object within one of the software elements;

a set of action nodes, each of which represents a functional object within one of the software elements that responds to control interactions and produces control interactions; and

a set of directed edges, each of which connect two nodes and represents implication between the two nodes.

2. A static error checking system according to claim 1 wherein each edge of the set of directed edges has an origin and a destination and only responds to a true value at the origin.

3. A static error checking system according to claim 1 wherein each edge of the set of directed edges has an origin and a destination and only responds to a false value at the origin.

4. A static error checking system according to claim 1 wherein each edge of the set of directed edges has an origin and a destination and only asserts a false value at the destination.

5. A static error checking system according to claim 1 wherein each edge of the set of directed edges has an origin and a destination and only asserts a true value at the destination.

6. A data structure for representing control constraints and control actions of a software system, the software system comprising at least two software elements with explicit control interactions between the software elements, the data structure comprising:

a set of conjunctive boolean guards on state changes within the software system;

a set of boolean guards on functional objects within the software elements;

a set of functional control objects within the software elements, each functional control object being responsive to a control interaction and capable of producing a control interaction; and

a set of relational connections between two elements from the set of conjunctive boolean guards, the set of boolean guards on objects, and the set of functional control objects, each pointer representing implication between the two elements it connects.

7. A data structure according to claim 6 wherein each edge of the set of directed edges has an origin and a destination and only responds to a true value at the origin.

8. A data structure according to claim 6 wherein each edge of the set of directed edges has an origin and a destination and only responds to a false value at the origin.

9. A data structure according to claim 6 wherein each edge of the set of directed edges has an origin and a destination and only asserts a false value at the destination.

10. A data structure according to claim 6 wherein each edge of the set of directed edges has an origin and a destination and only asserts a true value at the destination.

11. A static error checking system for debugging a software system, the software system comprising software elements which expose control interaction and data flow interactions between the software elements, by representing the control interactions and the data flow interactions in a graph, the graph comprising:

a set of conjunctive nodes, each of which represents a conjunctive boolean guard on state changes within the software system;

a set of disjunctive nodes, each of which represents a boolean guard on a functional object within one of the software elements;

a set of action nodes, each of which represents a functional object within one of the software elements that is responsive to a control interaction and capable of producing a control interaction;

a set of data flow nodes, each of which represents a data flow interaction between a first and a second software elements; and

a set of directed edges, each of which connects a first node to a second node and represents implication between the first and second nodes.

12. A static error checking system according to claim 11 wherein each edge of the set of directed edges has an origin and a destination and only responds to a true value at the origin.

13. A static error checking system according to claim 11 wherein each edge of the set of directed edges has an origin and a destination and only responds to a false value at the origin.

14. A static error checking system according to claim 11 wherein each edge of the set of directed edges has an origin and a destination and only asserts a false value at the destination.

15. A static error checking system according to claim 11 wherein each edge of the set of directed edges has an origin and a destination and only asserts a true value at the destination.

16. A data structure for representing control constraints and control actions of a software system, the software system comprising at least two software elements with explicit control interactions between the software elements, the data structure comprising:

a set of conjunctive boolean guards on state changes within the software system;

a set of boolean guards on functional objects within the software elements;

a set of functional control objects within the software elements, each functional control object being responsive to a control interaction and capable of producing a control interaction;

a set of data flow nodes, each of which represents a data flow interaction between a first and a second software elements; and

a set of relational connections between two elements from the set of conjunctive boolean guards, the set of boolean guards on objects, and the set of functional control objects, each pointer representing implication between the two elements it connects.

17. A data structure according to claim 16 wherein each edge of the set of directed edges has an origin and a destination and only responds to a true value at the origin.

18. A data structure according to claim 16 wherein each edge of the set of directed edges has an origin and a destination and only responds to a false value at the origin.

19. A data structure according to claim 16 wherein each edge of the set of directed edges has an origin and a destination and only asserts a false value at the destination.

20. A data structure according to claim 16 wherein each edge of the set of directed edges has an origin and a destination and only asserts a true value at the destination.

21. A method for converting a control graph representation of a software system, having a state space and an initial state, into a binary decision diagram of the software system comprising:

transforming the control graph to express a potential next state of the software system after a predetermined period of time; and

generating a binary decision diagram based on the transformed control graph, whereby known static error checking techniques may be used to further identify any unexpected behavior of the software system without incurring the cost of fully elaborating the state space of the software system.

22. A method according to claim 21 wherein transforming the control graph comprises unrolling the control graph.

23. A method according to claim 22 wherein generating the binary decision diagram comprises using the apply algorithm on a characteristic function of the unrolled control graph.

24. A method according to claim 23 wherein unrolling the control graph comprises:

creating a copy of each disjunctive node, each disjunctive node represents a boolean guard on a functional object within one of the software elements;

creating a copy of each conjunctive node, each conjunctive node represents a conjunctive boolean guard on state changes within the software system;

creating a copy of each action node, each action node represents a functional object within one of the software elements that is responsive to a control interaction and capable of producing a control interaction, if the functional object it represents performs a predetermined function without a predetermined delay;

for each delayed action node which represents a functional object within one of the software elements that has a predetermined delay in responding to. or producing, a control interaction, creating a sensing edge to connect the delayed action node to a corresponding node in the control graph representing the initial state of the system and creating an outgoing edge to connect the corresponding node, in the control graph representing the initial state of the system, to a corresponding next node, which represent the potential next state of the system;

for outgoing edge, that is also an event edge, connecting the outgoing edge to a create event disjunctive node, which represents an event generated by the corresponding node in the control graph representing the initial state of the system;

for each created event disjunctive node, creating an edge from the created event disjunctive node to an event conjunctive node;

for each event conjunctive node, creating an edge from the node that generated the event to the event conjunctive node, and creating an edge from the event conjunctive node to the copy of the node that generated the event.

25. A bit vector for use in debugging software systems, the software system comprising at least a first and second component and a coordinator for implementing a predetermined coordination scheme for managing control and dataflow interactions between the first and second components, the first and second components connected to the coordinator by a first and second pair of complimentary coordination interfaces, respectively, each coordination interface in the first pair of complimentary coordination interfaces comprising a control port for transferring control state between the respective component and the coordinator, each control port having a control state value representing on or off, the bit vector comprising:

one bit corresponding to each control port within the software system,

each bit having a boolean value representing the control state value of its corresponding control port at a predetermined time.